Via E-File

PATENT APPLICATION
Docket: 13768.1256

CERTIFICATE OF TRANSMISSION
I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted *via* the USPTO EFS Filing System on the date shown below to Mail Stop Appeal Brief-Patents Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Date:________________ /______________________________

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Appellants(s):	Anson Horton *et al.*	Examiner:	Cheneca Smith
Serial No:	10/808,905	Art Unit:	2192
Filing Date:	March 25, 2004	Conf. No:	7406

Title: ATTRIBUTED DEBUGGING

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Appellants' representatively submit this brief in connection with an appeal of the above-identified patent application. Payment is being submitted *via* credit card in connection with all fees due regarding this appeal brief. In the event any additional fees may be due and/or are not covered by the credit card, the Commissioner is authorized to charge such fees to Deposit Account No. 23-3178..

I. Real Party in Interest (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the subject application.

II. Related Appeals and Interferences (37 C.F.R. §41.37(c)(1)(ii))

Appellants, appellants' legal representative, and/or the assignee of the present application are not aware of any appeals or interferences which may be related to, will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §41.37(c)(1)(iii))

Claims 1-22 stand rejected by the Examiner. The rejection of claims 1-22 is being appealed.

IV. Status of Amendments (37 C.F.R. §41.37(c)(1)(iv))

No claim amendments were entered after the Final Office Action.

V. Summary of Claimed Subject Matter (37 C.F.R. §41.37(c)(1)(v))

Independent Claim 1

Independent claim 1 recites a computer-implemented attributed debugging system, comprising a debugger that facilitates debugging of a computer software application. (*See e.g.*, pg. 6, lns. 13-31, and FIG. 1, element 110.) The system also comprising an expression evaluator that evaluates an attribute associated with the computer software application according to an attribute definition, and presents debug information associated with the computer software application in accordance with the attribute definition (*see e.g.*, pg. 7, ln. 30 to pg. 8, ln. 4 and FIG. 1, element 120), wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format. (*See e.g.*, pg. 7, ln. 1 to ln. 28.)

Independent Claim 17

Independent claim 17 recites a computer-implemented method facilitating attributed debugging, comprising determining whether a process has an attribute attached (*See e.g.*, pg. 18, lns 1-3) and utilizing a definition of the attribute to display debug information, if the process has an attribute attached. (*See e.g.*, pg. 18, lns. 3-6.) The debug information is displayed in a developer-customizable format, the attribute includes a declarative indication of how to display the debug information. (*See e.g.*, pg. 2, lns. 25-29.)

Independent Claim 20

Independent claim 20 recites a data packet stored on computer readable media, the data packet transmitted between two or more computer components that facilitates debugging. (*See e.g.*, pg. 5, ln. 29 to pg. 6 ln. 2.) The data packet comprising an attribute, the attribute providing information associated with debugging of a computer software application (*see e.g.*, pg. 6, lns. 15-17), the information is presented in a developer-customizable format in accordance with the attribute information (*see e.g.*, pg. 7, ln. 1 to ln. 28).

Independent Claim 21

Independent claim 21 recites a computer readable medium storing computer executable components of an attributed debugging system comprising a debugger component that facilitates debugging of a computer software application (*see e.g.*, pg. 6, lns. 13-31, and FIG. 1, element 110) and an expression evaluator component that employs an attribute definition to evaluate an attribute associated with the computer software application (*see e.g.*, pg. 7, ln. 30 to pg. 8, ln. 4 and FIG. 1, element 120) to present debug information associated with the computer software application to a developer. The debug information is presented in a developer-customizable format and is presented in accordance with the attribute definition (*see e.g.*, pg. 7, ln. 1 to ln. 28).

Independent Claim 22

Independent claim 22 recites a computer-implemented attributed debugging system comprising means for storing an attribute associated with a computer software application (*see e.g.*, pg. 14, lns. 27-30, FIG. 2, element 220) and means for employing the stored attribute and an attribute definition to present debug information associated with the computer software application to a developer (*see e.g.*, pg. 14, lns. 27-32 and FIG. 2, element 120), wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format (*see e.g.*, pg. 7, ln. 1 to ln. 28).

The "means for" limitations described above are identified as limitations subject to the provisions of 35 U.S.C. §112 ¶6. The structures corresponding to these limitations are identified with reference to the specification and drawings in the above noted parentheticals.

VI. Grounds of Rejection to be Reviewed (37 C.F.R. §41.37(c)(1)(vi))

A. Whether claims 1, 5-7, 13-17 and 19-22 are unpatentable under 35 U.S.C. §103(a) over Bates *et al.* (U.S. Patent Appln. Pub. 2003/0221185) in view of Bates *et al.* (U.S. 7,251,808), hereinafter referred to as Bates_2.

B. Whether claims 2-4 and 8-12 are unpatentable under 35 U.S.C. §103(a) over Bates *et al.* (U.S. Patent Appln. Pub. 2003/0221185) in view of Bates *et al.* (U.S. 7,251,808), hereinafter referred to as Bates_2, and further in view of Dandoy (U.S. Appln. Pub. 2004/0230954).

VII. Argument (37 C.F.R. §41.37(c)(1)(vii))

A. <u>Rejection of Claims 1, 5-7, 13-17 and 19-22 Under 35 U.S.C. §103(a)</u>

Claims 1, 5-7, 13-17 and 19-22 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bates *et al.* (U.S. Patent Appln. Pub. 2003/0221185) in view of Bates *et al.* (U.S. 7,251,808), hereinafter referred to as Bates_2. This rejection should be reversed for at least the following reason. The combination of Bates *et al.* and Bates_2 does not teach nor suggest all elements of the subject claims.

<u>Claims 1, 5-7, and 13-16</u>

Independent claim 1, from which claim 5-7 and 13-16 depend recites *a computer-implemented attributed debugging system, comprising ...* ***an expression evaluator that evaluates an attribute*** *associated with the computer software application* ***according to an attribute definition and presents debug information*** *associated with the computer software application* ***in accordance with the attribute definition****, wherein* ***the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format****.* Attributes are keyword-like tags that specify additional information about entities. (*See e.g.*, pg. 2, lns. 6-7 and pg. 6, lns. 7-8). A debugging system can refer to the information in the attribute to determine how objects should be used. (*See e.g.*, pg. 2, lns. 8-10 and pg. 6, lns. 8-11.) Attributed debugging relates to a debuggee (*e.g.*, process to be debugged) that includes attribute(s) that can be employed to facilitate debugging. (*See e.g.*, pg. 2, lns. 22-24.) The attribute definition declaratively indicates how the debug information is presented and since attributes are an extensible, declarative manner of conveying runtime information, the attributes allow that behavior to be specified. (*See e.g.*, pg. 2, ln. 25 to pg. 3, ln. 23.) Attributed debugging allows a developer of a type, who also understood and specified the runtime behavior of that type, to specify how the type will appear when it is being debugged. (*See e.g.*, pg. 6, lns. 18-20.)

In an example, attributed debugging allows the manipulation of the view of data in the debugger by allowing annotations that can be controlled. (*See e.g.*, pg. 2, lns. 30-31). This can facilitate control of a value that should be shown at a top-level, which can mitigate the need to expand the object for additional information. (*See e.g.*, pg. 3, lns. 1-2.) This can also facilitate control of whether a field or property should be presented, provide more descriptive information, and whether a type should be shown fully expanded. (*See e.g.*, pg. 3, lns. 5, 10, and 12-13). Further, this can include control of a value that should be presented for a field or property, which can be based on a result of an evaluation of an expression. (*See e.g.*, pg. 3, lns. 7-8.) The cited art does not teach or suggest at least these elements.

Bates *et al.* relates to providing descriptions of variables and/or providing comments or other information associated with the variables available while debugging. (*See e.g.*, pg. 1, ¶[0003] and ¶[0009] and pg. 2, ¶[0022] and ¶[00025].) An "attribute" for the variable can be set to "off" or "on". (*See e.g.*, pg. 6, ¶[0064] and Fig 3, elements 312 to 322 and corresponding text at pg. 4, ¶[0047].) These attributes are "machine-generated comments" or "use information" and includes a group of characters representative of a word or words, such as G(global), S(static), I(index), P(parameter), R(return), and C(call). (*See e.g.*, pg. 2, ¶[0026].) If an attribute is set to "on", the appropriate indicator is associated with the variable value. (*See e.g.*, pg. 6, ¶[0064].) Bates *et al.* merely relates to a filtration system that allows certain data to be included or eliminated from view (by specifying whether that data is "on" or "off"). In other words, Bates *et al.* merely allows the user to decide which characters will be displayed (those for which the data is "on") and which will not be displayed (those for which the data is "off"). However, Bates *et al.* does not add anything to the debugging process, such as by evaluating an attribute to present debug information according to an attribute definition, which declaratively indicates ***how*** the debug information is presented in a developer-customizable format, as claimed.

The secondary art, Bates_2, is incorrectly relied upon to overcome the deficiencies of Bates *et al.* Specifically, the Examiner concedes in the Final Office Action that Bates *et al.* does not teach or suggest *wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format* and, thus, Bates_2 is relied upon. However, as discussed above Bates *et al.* does not teach or suggest the elements of independent claim 1 that recite an attribute and/or an attribute definition and Bates_2 does not overcome this deficiency of Bates *et al.* Bates_2 relates to a graphical debugger with a loadmap display manager and a custom record display manager that implements a custom record display function. (*See e.g.* col. 2, lns. 1-12.) A user can program the user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.*, col. 3, lns. 49-52.) When a variable of that type is encountered, the GUI initially displays only the user-programmed fields for the variable or the record. (*See e.g.*, col. 2, lns. 52-55.) Thus, Bates_2 combined with Bates *et al.* merely teaches that characters for the user-selected fields will be displayed on GUI, the organization of which can be customized by the user. However, Bates_2 in combination with Bates *et al.* does not teach or suggest *an expression evaluator that evaluates an attribute ...according to an attribute definition and presents debug information...in accordance with the attribute definition.* In other words, Bates_2 in combination with Bates *et al.* does not teach the ability to ***evaluate*** an attribute and, therefore, does not overcome at least this deficiency of Bates *et al.*

Based on at least the above, the combination of Bates *et al.* and Bates_2 does not teach or suggest all elements recited in independent claim 1, and therefore the claims that depend there from.

Claims 17 and 19

Independent claim 17, from which claim 19 depends, recites *a computer-implemented method facilitating attributed debugging, comprising determining whether a process has an attribute attached and* ***utilizing a definition of the attribute to display debug information****, if the process has an attribute attached, the debug information is displayed in a developer-customizable format,* ***the attribute includes a declarative indication of how to display the debug information****.* The attribute provides additional information about entities and the definition of the attribute declaratively indicates how to present the debug information. (*See e.g.*, pg. 2, lns. 6-7 and ln. 25 to pg. 3, ln. 23.) The cited art does not teach or suggest at least these features.

Bates *et al.* relates to machine-generated comments or use information that can be set to "off" or "on". (*See e.g.,* pg. 2, ¶[0022] and ¶[0026].) However, Bates *et al.* does not teach or suggest utilizing a definition of an attribute to display debug information, as claimed. Further, as conceded in the Final Office Action, Bates *et al.* does not teach or suggest *the attribute includes a declarative indication of how to display the debug information*, as claimed, and Bates_2 is relied upon to overcome this deficiency of Bates *et al.* Bates_2 relates to allowing a user to program a user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.,* col. 3, lns. 49-52.) However, Bates_2 does not teach or suggest utilizing a definition of an attribute to display debug information. Thus, the combination of Bates_2 and Bates *et al.* merely teaches that the user can indicate which attributes are to be sent to the GUI. Bates_2 and Bates *et al.* teach that the GUI governs the arrangement of the display of characters for the user-selected data. Nothing in the combination of Bates_2 and Bates *et al.* teaches or suggests that an aspect of the attribute indicates how to display the debug information and, therefore, this combination does not teach this limitation of claims 17 and 19. Thus, this rejection should be withdrawn.

Claim 20

Independent claim 20 recites *a data packet ... comprising an attribute, **the attribute providing information associated with debugging of a computer software application, the information is presented in a developer-customizable format in accordance with the attribute information***. The presentation of the information in accordance with the attribute information allows a developer of a type, who also understood and specified the runtime behavior of that type, to specify how the type will appear when it is being debugged. (*See e.g.,* pg. 6, lns. 18-20.) Neither Bates *et al.* nor Bates_2, considered alone and in combination, teach or suggest these elements of claim 20.

Bates *et al.* relates to machine-generated comments or use information that can be set to "off" or "on". (*See e.g.,* pg. 2, ¶[0022] and ¶[0026].) Bates_2 relates to allowing a user to program a user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.,* col. 3, lns. 49-52.) The combination of Bates *et al.* and Bates_2 does not teach or suggest ***an attribute providing information*** associated with debugging of a computer software application, as claimed. Thus, the combination of Bates *et al.* and Bates_2 does not teach or suggest each limitation of Claim 20, and this rejection should be removed.

Claim 21

Independent claim 21 recites *a computer readable medium ... comprising a debugger component and* ***an expression evaluator component that employs an attribute definition to evaluate an attribute*** *... to present debug information ...,* ***the debug information is presented in a developer-customizable format and is presented in accordance with the attribute definition.*** Presenting the debug information according to the attribute definition in a developer-customizable format provides a developer of a type, who also understood and specified the runtime behavior of that type, to specify how the type will appear when it is being debugged. (*See e.g.*, pg. 6, lns. 18-20.) Neither Bates *et al.* nor Bates_2 teach or suggest these elements.

Bates *et al.* discusses machine-generated comments (or use information) that can be set to "off" or "on". (*See e.g.*, pg. 2, ¶[0022] and ¶[0026].) Bates_2 discusses allowing a user to program a user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.*, col. 3, lns. 49-52.) However, the combination of the cited art does not teach or suggest an expression evaluator component that employs an attribute definition to evaluate an attribute ... the debug information is presented in a developer-customizable format and is presented in accordance with the attribute definition, as claimed. Thus, Bates *et al.* combined with Bates_2 does not teach or suggest each limitation of claim 21, and this rejection should respectfully be removed.

Claim 22

Independent claim 22 recites *a computer-implemented attributed debugging system comprising ... means for storing an attribute and* ***means for employing the stored attribute and an attribute definition to present debug information*** *...,* ***wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format.*** Presenting the debug information by employing a stored attribute and an attribute definition allows a developer of a type, who also understood and specified the runtime behavior of that type, to specify how the type will appear when it is being debugged. (*See e.g.*, pg. 6, lns. 18-20.) Neither Bates *et al.* nor Bates_2 teach or suggest these elements of claim 22.

Bates *et al.* relates to machine-generated comments or use information that can be set to "off" or "on". (*See e.g.*, pg. 2, ¶[0022] and ¶[0026].) Bates_2 relates to allowing a user to program a user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.*, col. 3, lns. 49-52.) However, the combination of Bates *et al.* and Bates_2 does not teach or suggest means for employing the stored attribute and an attribute definition to present debug information ... wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format, as claimed.

Based on the above, the combination of the cited art does not teach nor suggest all elements recited in claims 1, 5-7, 13-17, and 19-22. Accordingly, this rejection should be reversed.

B. Rejection of Claims 2-4 and 8-12 Under 35 U.S.C. §103(a)

Claims 2-4 and 8-12 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bates *et al.* (U.S. Patent Appln. Pub. 2003/0221185) in view of Bates *et al.* (U.S. 7,251,808), hereinafter referred to as Bates_2, and further in view of Dandoy (U.S. Appln. Pub. 2004/0230954). This rejection should be reversed for at least the following reasons. The combination of the cited art does not teach nor suggest all elements recited in the subject claims.

The subject claims depend from independent claim 1. As discussed above, the combination of Bates *et al.* and Bates_2 does not teach or suggest all elements of claim 1. Specifically Bates *et al.* and Bates_2 do not teach nor suggest ***an expression evaluator that evaluates an attribute ... according to an attribute definition and presents debug information ... in accordance with the attribute definition,*** *wherein* ***the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format.*** Further, Dandoy relates to debugging a user interface but does not overcome the deficiencies of Bates *et al.* and Bates_2.

Accordingly, the combination of the cited art does not teach or suggest all elements of independent claim 1 and, therefore, claims 2-4 and 8-12 that depend therefrom. Thus, this rejection also should respectfully be reversed.

C. Conclusion

For at least the above reasons, the claims currently under consideration are believed to be patentable over the cited art. Accordingly, it is respectfully requested that the rejections of claims 1-22 be reversed.

If any additional fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 23-3178.

Dated May 29, 2009.

Respectfully submitted,

/Chad E. Nydegger, Reg. # 61020/

RICK D. NYDEGGER
Registration No. 28,651
JENS C. JENKINS
Registration No. 44,803
CHAD E. NYDEGGER
Registration No. 61,020
Customer No. 47973

VIII. Claims Appendix (37 C.F.R. §41.37(c)(1)(viii))

1. (Previously Presented) A computer-implemented attributed debugging system, comprising:

a debugger that facilitates debugging of a computer software application; and,

an expression evaluator that evaluates an attribute associated with the computer software application according to an attribute definition, and presents debug information associated with the computer software application in accordance with the attribute definition, wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format.

2. (Original) The system of claim 1, the expression evaluator evaluates an expression associated with a particular programming language.

3. (Previously Presented) The system of claim 2, the programming language is at least one of C#, J# or Visual Basic.Net.

4. (Original) The system of claim 2, comprising a plurality of expression evaluators, each expression evaluator associated with a particular programming language.

5. (Original) The system of claim 1, the expression evaluator invokes an overridden implementation of a ToString method to facilitate presentation of debug information.

6. (Previously Presented) The system of claim 5, the expression evaluator employs a result of the overridden ToString method as a value to be displayed for an object, field, property, or combinations thereof.

7. (Original) The system of claim 1, the attribute employed to determine at least one of how and whether a type or member is displayed.

8. (Original) The system of claim 7, the attribute employing an enumeration.

9. (Original) The system of claim 8, one enumeration value associated with an indication that the type or member should not be displayed to the developer.

10. (Original) The system of claim 8, one enumeration value associated with an indication that if a type is hierarchical, it should be expanded by default.

11. (Original) The system of claim 8, one enumeration value associated with an indication that a type should not be expanded by default.

12. (Original) The system of claim 8, one enumeration value associated with an indication that a target element itself should not be shown, but should instead be automatically expanded to have its member(s) displayed.

13. (Original) The system of claim 1, the attribute employed to what is displayed for a class and/or field.

14. (Original) The system of claim 13, an argument to the attribute comprising a string that is displayed in a value column for an instance of the class and/or field.

15. (Previously Presented) The system of claim 14, the argument associated with one of a field, a property or a method, or combinations thereof.

16. (Previously Presented) The system of claim 1, further comprising an attribute cache directory that stores an attribute associated with the computer software application, the expression evaluator employing the stored attribute to present debug information.

17. (Previously Presented) A computer-implemented method facilitating attributed debugging, comprising:

determining whether a process has an attribute attached; and,

utilizing a definition of the attribute to display debug information, if the process has an attribute attached, the debug information is displayed in a developer-customizable format, the attribute includes a declarative indication of how to display the debug information.

18. (Original) The method of 17, further comprising at least one of the following:

determining whether a ToString method has been overridden; and,

invoking the overridden ToString method to facilitate debugging, if the ToString method has been overridden.

19. (Original) A computer readable medium having stored thereon computer executable instructions for carrying out the method of claim 17.

20. (Previously Presented) A data packet stored on computer readable media, the data packet transmitted between two or more computer components that facilitates debugging, the data packet comprising:

an attribute, the attribute providing information associated with debugging of a computer software application, the information is presented in a developer-customizable format in accordance with the attribute information.

21. (Previously Presented) A computer readable medium storing computer executable components of an attributed debugging system comprising:

a debugger component that facilitates debugging of a computer software application; and,

an expression evaluator component that employs an attribute definition to evaluate an attribute associated with the computer software application to present debug information associated with the computer software application to a developer, the debug information is presented in a developer-customizable format and is presented in accordance with the attribute definition.

22. (Previously Presented) A computer-implemented attributed debugging system comprising:

means for storing an attribute associated with a computer software application; and,

means for employing the stored attribute and an attribute definition to present debug information associated with the computer software application to a developer, wherein the attribute definition declaratively indicates how the debug information is presented in a developer-customizable format.

IX. Evidence Appendix (37 C.F.R. §41.37(c)(1)(ix))

None.

X. Related Proceedings Appendix (37 C.F.R. §41.37(c)(1)(x))

None.